2100030336

M. Gayathri Ankitha

.NET PROGRAMMING

LAB-8

IN-LAB

TASK1: Task **Delegates and events** refers to task **Collections**

To add the following new functionalities to the project created in task Collections:

Include two events in the **CustomArray** type:

- The **OnChangeElement** event occurs when the indexer changes the element value (if the old and new element values match, the event is not raised)
- The **OnChangeEqualElement** event occurs if a value equal to the index of the changed element is written to the element (if the old and new values of the element match, the event is not raised)

Use the **ArrayHandler** delegate to create the event.

The event handler takes two parameters: an object **sender** - a reference to the **CustomArray** instance that generated the event, and an event argument **ArrayEventArgs <T> e**. In the event argument, write in the **Id** field the index by which the user changes the element of the **CustomArray** array, in the **Value** field - the new value of the element by the Id index, in the **Message** field - an arbitrary string message

```
using System;
public class CustomArray<T>
{
    public delegate void ArrayHandler<T>(object sender, ArrayEventArgs<T> e);
    public event ArrayHandler<T> OnChangeElement;
    public event ArrayHandler<T> OnChangeEqualElement;
    private T[] array;
    private int startIndex;
    public CustomArray(int length, int startIndex)
    {
        if (length <= 0)
            throw new ArgumentException("Length must be greater than zero.");
        this.array = new T[length];
        this.startIndex = startIndex;
    }
    public CustomArray(params T[] values) : this(values.Length, 0)
    {
        Array.Copy(values, this.array, values.Length);
    }
    public int FirstIndex => startIndex;
    public int LastIndex => startIndex + array.Length - 1;
    public int Length => array.Length;
    public T this[int index]
    {
        get
```

```
        {
            if (!IsValidIndex(index))
                throw new IndexOutOfRangeException("Index is out of range.");
            return array[index - startIndex];
        }
        set
        {
            if (!IsValidIndex(index))
                throw new IndexOutOfRangeException("Index is out of range.");
            T oldValue = array[index - startIndex];
            array[index - startIndex] = value;
            if (!oldValue.Equals(value) && OnChangeElement != null)
            {
                OnChangeElement(this, new ArrayEventArgs<T>(index, value,
"Element changed"));
            }
            if (index.Equals(value) && OnChangeEqualElement != null)
            {
                OnChangeEqualElement(this, new ArrayEventArgs<T>(index, value,
"Element changed to index value"));
            }
        }
    }
    public T[] ToStandardArray()
    {
        return array;
    }

    private bool IsValidIndex(int index)
    {
        return index >= startIndex && index < startIndex + array.Length;
    }
}
public class ArrayEventArgs<T> : EventArgs
{
    public int Id { get; }
    public T Value { get; }
    public string Message { get; }
    public ArrayEventArgs(int id, T value, string message)
    {
        Id = id;
        Value = value;
        Message = message;
    }
}
class Program
{
    static void Main(string[] args)
    {
        try
        {
            CustomArray<int> intArray = new CustomArray<int>(1, 2, 3, 4, 5);

            intArray.OnChangeElement += HandleChangeElement;
            intArray.OnChangeEqualElement += HandleChangeEqualElement;
            intArray[2] = 10;
            intArray[3] = 3;
            intArray[4] = 4;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
```

```
    }
    static void HandleChangeElement(object sender, ArrayEventArgs<int> e)
    {
        Console.WriteLine($"Element at index {e.Id} changed to {e.Value}:
{e.Message}");
    }
    static void HandleChangeEqualElement(object sender, ArrayEventArgs<int> e)
    {
        Console.WriteLine($"Element at index {e.Id} changed to index value
{e.Value}: {e.Message}");
    }

}
```